Implementing Digital Twins of Smart Factories with Interval Algebra

Piotr Dziurzanski[1], Jerry Swan[1], Leandro Soares Indrusiak[1] and J. M. Ramos[2]
[1]University of York, Deramore Lane, York, YO10 5GH, UK. {firstname.lastname}@york.ac.uk
[2]ONA Electroerosion S.A., Barrio de Eguzkitza 1,48200 Durango, Spain

***Abstract*— This paper presents an interval algebra that can be used to create an analytic description of a smart factory. Such a description (recently termed a 'digital twin' of the factory) is used to evaluate alternative manufacturing configurations as part of a search-based optimisation process. Several extensions are proposed to the interval algebra for specifying smart factory production line details. A number of real-life manufacturing scenarios are described, related to Wire-cut Electrical Discharge Machining. The experimental results show the applicability and scalability of the proposed method.**

I. Introduction

With myriad practical applications, the Job-shop Scheduling Problem (JSP) is arguably one of the most widely studied optimisation problems [1]. In the JSP, jobs are assigned to resources at particular times in order to optimise certain key objectives, such as makespan or total workload of machines. However, the classic JSP and its popular extensions do not scale well to the problem sizes found in industry [2]. Due to the NP-hard nature of the problem, exact solutions are not generally possible for real-world problems. In practice, relatively simple dispatch heuristics are often used, where each temporal property (arrival time, processing time etc) is characterised with some degree of uncertainty [3]. Such heuristics assign a priority to each manufacturing process queued in each machine, which are then used to fetch these processes from the queue and thus determine the processing order [4]. Such rules are particularly applicable to the Dynamic Job-shop Scheduling problem (DJSP), in which manufacturing jobs are released over time, in contrast to typical JSP, when the whole jobset is known *a priori* [3]. When exact algorithms are applied to the DJSP, each new release of a job requires re-execution of the costly scheduling algorithm. Similarly, a new schedule needs to be determined when any process behaves different than expected (e.g. needs longer processing) or some unexpected event occurred in a plant (e.g. a machine failure). Since the scheduling process must respond to these anomalies, this approach is named *reactive scheduling* [5]. A number of heuristic approaches has been proposed to find a quasi-optimal schedule in reasonable time for DJSP [6]. Recently, a method for trading schedule optimisation time against expected schedule quality was proposed [7]. When the scheduling algorithm is responsive enough, it is possible to schedule in reaction to each unexpected event, assigning priorities to each manufacturing job explicitly, rather than craft more general and thus less effective dispatching rules [8]. By this means, a problem-specific job dispatching (schedule) is determined using a *hybrid modeling approach*, when a metaheuristic optimisation routine is coupled with a simulation or analytical model of a plant. The metaheuristic optimiser generates a set of candidate solutions evaluated by the plant model. Since that model mimics the physical plant behaviour, it is often referred to as its *digital twin* [9]. Since a digital twin of a plant respects all known manufacturing constraints, it always yields feasible solutions [8].

Regarding the two components of the hybrid modeling approach, metaheuristic optimisation routines have been widely analysed and surveyed, for example in [1]. The creation of adequate digital twins has attracted far less attention. Usually, general-purpose discrete-event simulation software is used, as surveyed in details in the 'Related Work' Section. However, some algebraic formalities, such as max-plus algebra [10] or interval algebra [11], are extensible enough to satisfy all the requirements imposed by the problem presented in this paper. In particular, this paper reviews and extends interval algebra. The applicability of the proposed method is demonstrated on real-world industrial scenarios.

The main contribution of this paper is the extension of the interval algebra. It extends previous applications of interval algebra to scheduling in many-core real-time systems [11], [12] into a formalism capable of acting as a digital twin of smart plants, characterised with sophisticated spatial and temporal dependences between resources and manufacturing stages. A number of real-life scenarios are presented, described and evaluated to highlight some possible applications of the proposed method.

The rest of the paper is organised as follows. After the brief survey of the related works in Section II, the problem to be solved is described in Section III. The interval algebra from [11] is reviewed and extended in Section IV and Section V, respectively. Real-world use cases are presented in Section VI, followed by conclusions in Section VII.

II. Related work

Prior to the introduction of explicit terminology for digital twins, the underlying notion was the *de-facto* approach of manufacturing optimisation via simulation models of a plant. Law and McComas describe an early hybrid of search methods with discrete-event simulation models for manufacturing. Even at that early phase of the hybrid modeling approach, such meta-heuristics as evolution strategies, taboo search, neural networks or simulated annealing have been available. However, the applied metaheuristics have not changed greatly since then. In particular, two different optimisation

packages have been applied there to one small practical problem and the obtained results have been promising enough to conclude that the practice of simulation-based optimisation would grow significantly. This prophecy has fulfilled since and especially recent years yielded in numerous related publications [13]. For example, in [8], a simulation model has been built using a discrete-event simulation software named Anylogic [14], a commercial tool to simulate complex business systems. A plant can be modeled using process flowcharts, statecharts, action charts or stock & flow diagrams, integrated with 2D and 3D visualisers. Although useful in its application domain, it is not open and operates at too coarse scale to be applied in the flow proposed in this paper, not mentioning its commercial license. In a similar way, Klemmt describes a discrete event simulation system from the Simcron company named Modeller [15]. Klemmt stresses that the number of available simulation elements is limited, but thanks to the scripting capabilities even complex manufacturing systems could be modeled. The simulation engine speed of the GUI-less version is reportedly sufficient to perform multiple simulations as required by heuristic optimisation algorithms. A couple of plant features found in real plants was implemented, for example release dates, due dates, branches or setup times. However, it does not appear to be possible to model some plant features required by the real-world use cases presented in this paper, such as various machine operating modes or job preemptions.

Zho et al [16] applied the popular simulation software ARENA©implementing complex simulation logic and controlling data flow via the Visual Basic for Applications (VBA) environment embedded in that tool. A Multi-Objective Genetic Algorithm (MOGA) was employed to find Pareto-optimal solutions in terms of economic performance and green yield measures.

Despite being so widely applied, simulation-based hybrid systems are characterised with lengthy response times in comparison with analytical models [17]. Simulation models are also often characterised with the lack of clear structure and often cannot reach an acceptable level of performance [18]. In contrast, an advantage of analytical techniques is the ease of computing using explicit mathematical formulas and numerical computation methods [19]. As observed by Shao et al [20], simulation-based evaluations are customised for special purposes and are difficult to apply them to other scenarios. Although all these drawbacks are significant, applying analytical techniques as alternative way of performance evaluation seems to be far less popular in manufacturing optimisation. This is in contrast with other application domains, e.g. performance evaluation of complex computing systems, where analytical methods are used predominantly [21]. Nevertheless, a number of analytical alternatives have been proposed. For example, in [22] a rather simple 3-step objective function evaluation algorithm has been described. The evaluation technique proposed in that paper does not consider such features as multi-modal resource behaviour or multi-objective optimisation, but the authors stress that these features could be added in future. Some techniques popular in other domains are more expressive. One such technique is Network calculus, a theory of deterministic queuing systems proposed by Cruz [23]. It offers an alternative approach to queuing theory, using upper bounds to characterize arrivals and lower bounds to describe services. Using this approach, bounds can be easily computed for network performance metrics such as delay or backlog. Since this formalism is aimed at computer networks, a considerable modification would be needed to consider all the requirements imposed by the problem considered in this paper.

From this literature survey, it may be concluded that hybrid modeling approach coupled with analytical evaluation is not well explored in the field of smart factories and that applying extended formalities known from other fields can be promising.

The formalism of interval algebra [11], previously used for computer-system resource scheduling, is more general and expressive enough to deal with the majority of these requirements. As it is shown in this paper, this algebra can be extended to express the remaining requirements in a relatively simple way.

III. Problem formulation

The problem considered in this paper is an extended version of the classic flexible Job-shop Scheduling Problem (EFJSP), in which a set of n jobs needs to be processed in a plant equipped with m machines so that the makespan or financial cost is minimised. The set of jobs is denoted as Γ, $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. The set of machines is denoted as Π, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$.
The problem has the following constraints:

1) There are different types of machines in a plant, $\Pi_{\alpha 1}, \ldots, \Pi_{\alpha t}$.
2) Each job τ_i can only be executed by a subset of machines in a plant $\Pi_i \subset \Pi$, possibly of various types.
3) The processing of a job may require it to be allocated to more than one machine.
4) Two jobs τ_{i1} and τ_{i2} can be dependent or directly dependent on each other. The dependency relation denotes that τ_{i2} cannot start before τ_{i1} finishes, whereas the direct dependency requires that τ_{i2} starts exactly when τ_{i1} finishes.
5) Each job can be preempted at certain time points.
6) Machine π_j can operate in one of a set of modes, each mode differing in processing time and economical costs.
7) Some machines π_{j1} and π_{j2} cannot be used at the same time.
8) Allocation of job τ_{i1} to a certain machine can decrease the set of machines which can process the jobs dependent on τ_{i1}.
9) Certain sequences of two jobs τ_{i1} and τ_{i2} scheduled to be processed subsequently by the same machine can require a time gap of certain length between them (corresponding to e.g. cleaning the machine in a physical plant).

The analytical formalism used for makespan evaluation of EFJSP needs to be expressive enough to consider all these assumptions. In the following section, the principles of interval algebra [12] are presented, followed by the necessary extensions to comply with the above constraints.

IV. Interval algebra for real-time scheduling

Interval algebra (IA) was introduced by Indrusiak et al [12] for task scheduling in real-time computer-based systems. To handle the constraints associated with EFJSP, some extensions are required. For the sake of self-consistency of this paper, the most important features of IA are described in this section, followed by the required new features presented in the subsequent section.

In EFJSP, a manufacturing process is viewed as a set of jobs $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots\}$. Jobs which appear exactly once during the whole considered time horizon are often referred to as singletons. Periodic or sporadic jobs can be treated as an infinite series of singleton occurrences that are released periodically or less often than the provided inter-release time, respectively. Such jobs can represent e.g. a periodic maintenance operation of a resource. The j-th occurrence of a periodic or sporadic job τ_i is denoted with $\tau_{i,j}$. Jobs are mapped to plant resources, such as conveyors or machines, using a selected scheduler (e.g. FIFO or priority-preemptive).

In IA, jobs are represented using tuples that include their name, release time or task dependencies, and the interval length. In the assumed notation, each tuple element starts with symbol $\#$. The simplest case, a singleton job, can be represented by the time interval it requires from a notional resource as follows:

$$\#\tau_1\#0\#50, \tag{1}$$

where the first element of the tuple is a unique job identifier, the second is a non-negative real number representing the release time of the job and the third is a positive real number representing the actual length of the time interval. In the example above, job τ_1 is released at time 0 and requires 50 time units of a resource. Using traditional mathematical notation, this interval can be written as left-closed right-open bounded interval $[0, 50)$.

Using the notation shown in formula (1), any singleton job can be expressed. A set of such intervals can represent independent jobs. An indirect dependency between two jobs τ_1 and τ_2, i.e., the situation when τ_2 can start only after τ_1 has been executed, is denoted with a job identifier as the second element of a tuple, instead of the release time of a job:

$$\#\tau_2\#\tau_1\#100. \tag{2}$$

The extension of IA presented in the following section can also express other inter-job relations found in industrial plants.

The notation shown in formula (2) is capable of denoting single dependency jobs. Multiple dependencies are specified as a dependency set:

$$\#\tau_3\#\{\tau_1, \tau_2\}\#150. \tag{3}$$

This notation assumes that the starting point of the interval corresponding to job τ_3 lies not earlier than at the highest endpoint among all the intervals it depends on. In this example, assuming that jobs τ_1 and τ_2 are defined as in formulæ (1) and (2), this leads to: $\tau_1 = [0, 50)$, $\tau_2 = [50, 150)$, $\tau_3 = [150, 300)$.

The intervals described with formulæ (1) - (3) are single-appearance and have a fixed release time. In case of a strictly periodic job, its instances are released every fixed amount of time, known as a period. Such a job occurrence series is denoted with the notation exemplified below, which is exactly the same as the notation of a singleton followed by the period (equal to 100 in this case):

$$\#\tau_4\#0\#20\#100. \tag{4}$$

Mathematically, formula (4) represents an infinite series of intervals, such as: $\tau_4 = [0, 20), [100, 120), [200, 220), \ldots$. The release time of sporadic jobs is not deterministic but has well defined the minimal time between two consecutive releases. In case of aperiodic jobs, those bounds do not exist. To model those cases, the release times can be represented with so-called aleatory variables. Those variables are associated with probability distributions that can constrain assumed values. More details on them can be found in Indrusiak et al [12].

A resource can be represented by an algebraic operation over all the jobs mapped onto it, each represented by its respective interval, corresponding to a selected dispatching heuristic γ. The algebraic operation determines how the resource is shared between the jobs mapped to it, and how the sharing affects their timings. In the used notation, the dispatching heuristic γ acts as an operator that is applied to the set of intervals surrounded by brackets

$$\gamma(\#\tau_1\#0\#50). \tag{5}$$

If more than one job is presented to a dispatching heuristic, the corresponding intervals are separated with a comma, as shown below

$$\begin{gathered}\gamma(\#\tau_1\#0\#50, \#\tau_2\#0\#100) = \\ \gamma(\#\tau_1\&50, \#\tau_2\&150) = \gamma([0, 150)),\end{gathered} \tag{6}$$

where γ is realised with the FIFO dispatching heuristic. In this example, two different ways to evaluate operator γ are presented. The first evaluation of the operator preserves the identities of the mapped jobs, and it indicates the completion times of each one of them after the symbol $\&$. This type of evaluation is referred to as information-preserving (or simply preserving). The second way to evaluate the operator is equivalent to the first, but it does not preserve any information about the individual operands. It simply determines the busy period(s) of the resource with one or more intervals. This type of evaluation is referred to as information-collapsing (or simply collapsing).

One of the crucial properties of each job is its *affinity*, which means that the job can be processed only by the designated resources. The job that can be executed on any resource available in a system is referred to as an untyped

job. All the earlier examples (1) - (6) presented untyped jobs only. If a job can be executed only on a single type of resources, it is a single-typed job. A multi-typed job can be executed on several (enumerated) resource types, possibly with different processing time. To describe a single-typed or multi-typed job, the notation supports the definition of different types of resources and different types of resource affinity. This can be expressed as follows, where each scalar in pointy brackets denotes a different type and the absence of type constraints implies untyped jobs or resources

$$\gamma(\#\tau_{10} < \Pi_{\alpha 1} > \#0\#15, \\ \#\tau_{11} < \Pi_{\alpha 2}, \Pi_{\alpha 3}, \Pi_{\alpha 8} > \#0\#20, \#\tau_{12}\#0\#14). \quad (7)$$

By allowing the definition of resource types and resource requirements, it is also possible to present transport jobs between two machines (e.g. using a conveyor) by modelling the job as two fully dependent intervals with distinct resource requirements, e.g. a resource of type $\Pi_{\alpha 1}$ for processing and $\Pi_{\alpha 2}$ for transporting. In this situation, the job can only be connected over the resource of type $\Pi_{\alpha 2}$ once it has finished being processed by resource belonging to $\Pi_{\alpha 1}$, which can be described as

$$\#\tau_{13} < \Pi_{\alpha 1} > \#0\#14, \quad (8)$$

$$\#\tau_{14} < \Pi_{\alpha 2} > \#\tau_{13}\#340. \quad (9)$$

The representation of load as an interval length, denoted by a positive real number, is already capable of representing a fixed load. To represent a typed fixed load, the specification of different interval lengths for different resource types uses a similar notation as the one introduced in formula (7), namely

$$\#\tau_{15} < \Pi_2, \Pi_4, \Pi_6 > \#0\# < 10, 20, 20 > . \quad (10)$$

Aleatory variables can be used to represent a probabilistic load or typed probabilistic load for both typed and untyped jobs. The details of stochastic interval algebra can be found in Indrusiak et al [12].

V. Interval algebra extension for EFJSP

When applied to industrial plants, IA should be capable of solving job scheduling problems with different manufacturing process topologies, including both single and multiple stages. In the latter, commodities can be manufactured on a range of alternative resources using different routes. This implies that additional relations have to be defined between certain resources, describing their affinity or anti-affinity. Another requirement is related to changeovers which can be sequence-dependent, requiring that the feature of sequence-dependent setup should be introduced. Another required feature stems from the material transfer requirements, as in a plant it is possible that certain jobs have to be executed immediately one after another. As a result, the general precedence relationship natively supported by IA needs to be extended to distinguish immediate precedence relationships. The new features required are described below.

A. Mutual exclusiveness of resources

In plants, certain resources cannot be used at the same time. For example, two conveyors may transport a raw material from different silos to the same weighing scale. To prevent resources π_1 and π_2 being active at the same time, it is necessary to define a mutex (mutual exclusion) relation between them. This relation is symmetric and transitive.

B. Different routes

In the multiproduct topology, different routes between processing machines are possible. However, equipment connectivity can be limited (partial), for example defined by existing conveyors or pipes. It is therefore necessary to constrain which resources can be used sequentially by jobs belonging to a single manufacturing. Two relations are introduced:

- An *affinity* relation between resource π_1 and resource π_2 means that for two different jobs τ_1 and τ_2 realising the same manufacturing order and $\tau_1 < \tau_2$ with respect to the topological order, where π_1 is compatible with job τ_1 and τ_2 is compatible with resource π_2, if τ_1 is allocated to π_1 then τ_2 can be allocated to π_2.
- An *anti-affinity* relation between resource π_1 and resource π_2. For two different jobs τ_1 and τ_2 associated with the same manufacturing order such that $\tau_1 < \tau_2$ with respect to the topological order, where π_1 is compatible with job τ_1 and π_2 is compatible with resource π_2, then if τ_1 is allocated to π_1 then τ_2 cannot be allocated to π_2.

C. Sequence-dependent setup

In real-world scheduling problems, changeovers can be sequence dependent [24]. An example is when in the same resource π_1 (e.g. mixer) two jobs τ_1 and τ_2 are to be processed one after another, each belonging to a different manufacturing order producing a different commodity (e.g. different colour paint). In such situations, an additional job τ_3 has to be processed by π_1 between τ_1 and τ_2 (e.g. cleaning of the mixer). In IA, a new function has to be introduced that takes as parameters: a resource, the job currently processed by that resource, the job subsequently to be processed by that resource. This function returns the job to be processed by π_1 between τ_1 and τ_2. This function returns the empty job (i.e. an untyped job with processing time 0) if the sequence-dependent setup is not defined for given parameters.

D. Intra-job relations

In the original interval algebra formulation, only one relation between jobs was defined, namely a general precedence relationship. In order to apply IA to EFJSP problems, more intra-job relations have to be considered. Seven relations between intervals were famously described by Allen [25], as summarised in Table I. One of these relations, y meets z, can be used to describe immediate precedence relationships between jobs associated with to the same manufacturing order. To increase the genericity of the fitness function evaluation block, all relations from Allen's interval algebra has been added to IA. These relations can be described

similarly to single dependency in formula (2), but using an appropriate relation symbol in from of the first operator, for example τ_1 m τ_2 can be specified as

$$\#\tau_2\#\mathsf{m}\{\tau_1\}\#50, \tag{11}$$

where the processing time of τ_2 equals 50 time units.

VI. Real-world use cases

The considered real-world use case is based on the discrete manufacturing process of Wire Electrical discharge machining (WEDM), in which a thermo-electric sparking process removes material using a wire to cut the desired shape of a part. Complex profiles with tight tolerances in hard conductive materials can be obtained. Minimising the cost per part while maintaining the required quality is the key objective.

The wire is unwound from a spool and it is the most expensive consumable in the process, so to select the optimum wire type is economically essential. Similarly, the cost per part can be decreased significantly with the right choice of the machine type and, if possible, applying 'eco mode' functionalities.

In the first considered scenario, a plant includes three WEDM machines suitable for various sizes of parts ('small', 'medium' and 'large'). Small workpieces can be manufactured on any machine size, medium parts require a medium or large machine, and large parts can be set up only on large machines. The cost of machine usage increases with the machine size within a series.

The processing time for each machine can be computed by considering the total profile length for the sequence of cuts required by the quality specification. The process parameters set for each cut in the sequence is selected from a data base (WEDM 'technology') indexed by the type of wire and the material of the part. The commodity cost is determined from the wire speed together with the wire price.

All parts can be manufactured in one from four ways named Manufacturing Ways (MWs). MW 1 denotes 0.25mm brass wire and the standard mode of the machine, whereas MW 2 assumes the same wire type and the eco mode of the machine. MW 3 and MW 4 denotes technologies employing 0.25 mm brass coated and copper wires, respectively. Considering (approximate) part processing time provided by a business partner and assuming market wire costs and the real wire consumption, the total wire cost for manufacturing each part can be estimated as the cost of using a machine of a particular size during the computed total manufacturing time. This way, the total manufacturing cost can be computed and used for the optimisation purposes.

To demonstrate the IA extensions proposed earlier in this paper, four real-world-based manufacturing scenarios are presented. The first of them demonstrates job allocation and scheduling via a multi-objective optimisation process. To model various modes (manufacturing ways) of the machines, separate abstract machines are created for each mode of a certain machine and mutual exclusiveness of resources

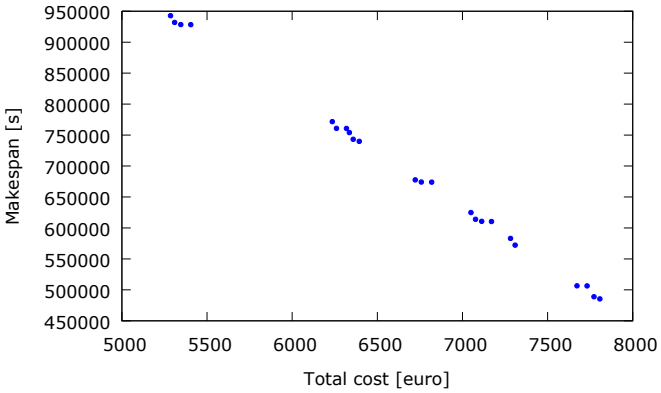


Fig. 1. Pareto front for scenario with manufacturing 6 parts using three machines of different size

is used to prevent concurrent allocation to various abstract machines corresponding with the same physical machine. In the second scenario, intra-job relations and sequence-dependent setups are applied. The third scenario investigates the extensions related to different routes and direct dependency in the manufacturing process. The final scenario demonstrates the scalability of the proposed approach.

In the first example scenario, it is assumed that there is only one machine of each size in the plant. Two small and four medium parts have been ordered. All parts are ready to be processed at any time of the analysed time horizon.

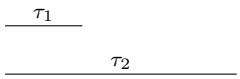
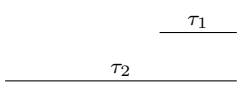
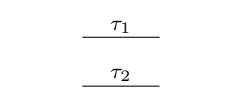
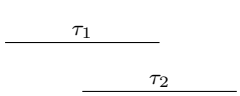
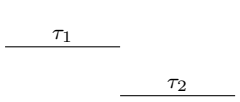
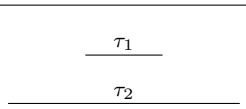
The trade-off between two conflicting objectives, makespan and cost per part, has been investigated for this example scenario and the associated Pareto front visualised in Fig. 1. All the solutions in the Pareto front are not *dominated* by any other solution, where solution M dominates solution N if M is at least as good as N in all objectives, and superior to N in at least one objective.

Presented with the Pareto front, it is therefore possible for the end-user to choose the solution that is most suitable, based on business priorities. For example, the leftmost point (which favors cost) allocates work to medium sized machines 75% of the time and small machines 25%. In contrast, the rightmost point (which favors makespan) assigns equally to all three machine sizes.

In the second scenario, four large parts have been ordered. These parts have different priorities and thus earlier manufacturing of the higher-priority parts is recommended. As some of the parts are not ready at the beginning of the schedule, a manufacturing of a lower-priority part can be performed when a job of a higher-priority part manufacturing is released. In this situation, the currently performed job can be preempted at certain moments, when a certain cut of the WEDM process is finished. Such manufacturing segmentation can be modeled with a sequence of IA jobs, with each (except the initial one) dependent on the previous. A job preemption, which involves removing of the currently manufactured part from a machine and installing a new one takes some time. This is modeled with an appropriate sequence-dependent setup, one of the IA extensions de-

TABLE I

BINARY RELATIONS IN ALLEN'S INTERVAL ALGEBRA

Relation	Symbol	Equivalent relation on endpoints	Illustration
τ_2 earlier than τ_1	$\tau_2 < \tau_1$	$\tau_2+ < \tau_1$	
τ_2 since τ_1	τ_2 s τ_1	$(\tau_2- = \tau_1-) \wedge (\tau_1+ < \tau_2+)$	
τ_2 finish τ_1	τ_2 f τ_1	$(\tau_2+ = \tau_1+) \wedge (\tau_2- < \tau_1-)$	
τ_2 equals τ_1	$\tau_2 = \tau_1$	$(\tau_2- = \tau_1-) \wedge (\tau_2+ = \tau_1+)$	
τ_2 overlaps τ_1	τ_2 o τ_1	$(\tau_2- < \tau_1-) \wedge (\tau_2+ > \tau_1-) \wedge (\tau_2+ < \tau_1+)$	
τ_2 meets τ_1	τ_2 m τ_1	$\tau_1+ = \tau_2-$	
τ_2 during τ_1	τ_2 d τ_1	$((\tau_2- > \tau_1-) \wedge (\tau_2+ = (\tau_1+)) \vee ((\tau_2- >= \tau_1-) \wedge (\tau_2+ < \tau_1+))$	

scribed earlier in this paper. In this scenario, the jobs are ready to be manufactured in the decreasing order of priorities. It means that at the beginning only the lowest-priority part is ready to be processed and thus the manufacturing process is initiated. After some time a higher-priority part is ready to be produced. At this moment, two manufacturing decisions are possible. Either the manufacturing of the lower-priority part manufacturing is preempted and the higher-priority part begins to be manufactured, or the higher-priority part manufacturing is suspended until the lower-priority part is finished. The first strategy decreases the makespan, as there is no time-consuming preemption modeled with a sequence-dependent setup, but the part finishing order violates the order as determined by the parts' priorities. There are hence two conflicting objectives: the makespan and the number of the given order priority violations. As shown in Fig. 2, during the optimisation process three alternative solutions have been found for this scenario. The makespan is inversely proportional to the number of priority order violations, since each preemption inserted some time gap to change the part in the machine. The leftmost Pareto-optimal point corresponds to the solution in which the parts are finished in the preferred order, but imposing that order requires two preemptions. The middle Pareto-optimal point corresponds to a solution with one preemption, but using this schedule one part with a lower-priority is finished earlier than a higher-priority part. Finally, the rightmost Pareto-optimal point corresponds to a solution resulting with the fastest manufacturing without preemptions, but two lower-priority parts are finished before other parts with higher priorities.

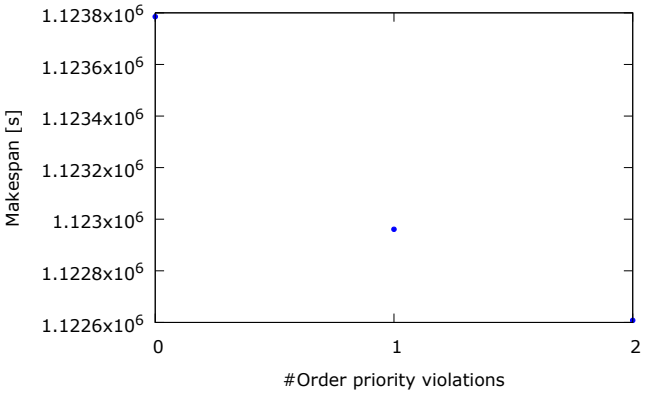


Fig. 2. Pareto front for scenario with preemption

In the third scenario, the manufacturing of each each part requires two stages. The first of them is performed

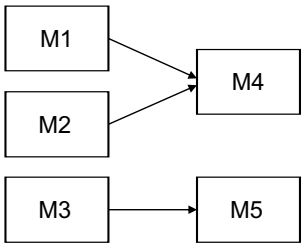


Fig. 3. An example plant architecture

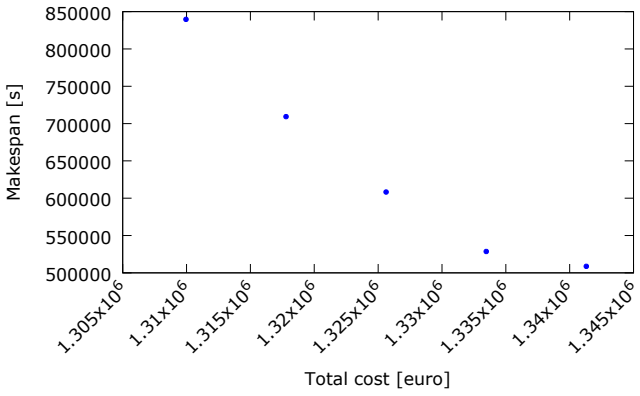


Fig. 4. Pareto front for scenario with two manufacturing stages

on machines M1-M3, whereas the second one requires M4 or M5. Each part can be manufactured on any machine. However, due to the plant architecture shown if Fig. 3, M4 can be used only when the first manufacturing stage has been performed on M1 or M2, whereas M5 can be used only after M3. These dependencies have been modelled with the affinity relations described earlier in this paper. As the part needs to be manufactured immediately at the second stage after stage 1 (there is no possibility of storing unfinished parts), the direct dependency (modelled by the intra-job 'meet' relational constraint) is used rather than the general dependency used in the previous scenario. The optimisation has been performed for two objectives, makespan and total cost. The Pareto-front for manufacturing 10 parts is presented in Fig 4.

The final scenario consists of a much larger plant and order and aims to show the scalability of the proposed optimisation method. Fig. 5 shows the optimisation time for tasksets ranging from 20 to 200 jobs. Despite the fact that above-linear complexity can be observed, the optimisation time even for the largest considered cases is close to 10 minutes, which is at least 3 orders of magnitude lower than the corresponding makespans and acceptable for the industrial partner. In case of shorter deadlines, value-based optimisation stopping criteria presented in [7] may be applied.

VII. Concluding remark

In this paper, a previous formulation of an interval algebra is extended to create a novel analytic description (or 'digital twin') of a smart factory. The extended interval algebra is used to model a real-life manufacturing application Wire-cut Electrical Discharge Machining. In contrast to previous work, the extended interval algebra models described constraints such as mutual exclusion and (anti-)affinity of resources, together with sequence-dependent resource allocation and temporal ordering relations. These constraints are used to model 4 real-world manufacturing scenarios. The schedules obtained via (multi-objective) metaheuristic optimisation are demonstrated to scale process 200 jobs in about 10 minutes — well beyond the current requirements of the industrial partner.

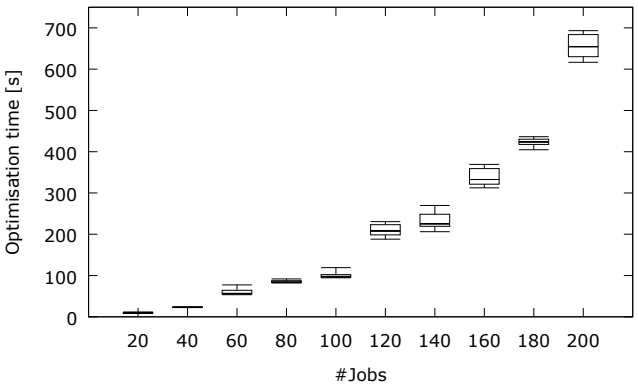


Fig. 5. Optimisation time for assorted job number in an example plant

Acknowledgement

The authors acknowledge the support of the EU H2020 SAFIRE project (Ref. 723634).

References

[1] C. I. Ali and K. A. Ali, "A research survey: review of flexible job shop scheduling techniques," *International Transactions in Operational Research*, vol. 23, no. 3, pp. 551–591. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12199

[2] K. Wang and S. Choi, "A holonic approach to flexible flow shop scheduling under stochastic processing times," *Computers & Operations Research*, vol. 43, pp. 157 – 168, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054813002839

[3] S. R. Lawrence and E. C. Sewell, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain," *Journal of Operations Management*, vol. 15, no. 1, pp. 71 – 82, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0272696396000903

[4] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: A comparison of rule representations," *Evol. Comput.*, vol. 23, no. 2, pp. 249–277, Jun. 2015.

[5] D. Karunakaran, Y. Mei, G. Chen, and M. Zhang, "Toward evolving dispatching rules for dynamic job shop scheduling under uncertainty," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 282–289. [Online]. Available: http://doi.acm.org/10.1145/3071178.3071202

[6] A. Ali, P. Hackney, D. Bell, and M. Birkett, "Genetic algorithms for solving bicriteria dynamic job shop scheduling problems with alternative routes," in *Proceedings of the The International Conference on Engineering & MIS 2015*, ser. ICEMIS '15. New York, NY, USA: ACM, 2015, pp. 31:1–31:8. [Online]. Available: http://doi.acm.org/10.1145/2832987.2833038

[7] P. Dziurzanski, J. Swan, and L. S. Indrusiak, "Value-based manufacturing optimisation in serverless clouds for industry 4.0," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 1222–1229. [Online]. Available: http://doi.acm.org/10.1145/3205455.3205501

[8] K. Kulkarni and J. Venkateswaran, "Hybrid approach using simulation-based optimisation for job shop scheduling problems," *Journal of Simulation*, vol. 9, no. 4, pp. 312–324, 2015. [Online]. Available: https://doi.org/10.1057/jos.2014.40

[9] M. Grieves and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. Cham: Springer International Publishing, 2017, pp. 85–113. [Online]. Available: https://doi.org/10.1007/978-3-319-38756-7_4

[10] P. Dziurzanski and L. Indrusiak, "Value-based allocation of docker containers," in *Parallel, Distributed and Network-Based Processing (PDP), 26th Euromicro International Conference on*. IEEE, 2018.

[11] L. S. Indrusiak and P. Dziurzanski, "An interval algebra for multiprocessor resource allocation," in *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, July 2015, pp. 165–172.

[12] L. Indrusiak, P. Dziurzanski, and A. Singh, *Dynamic Resource Allocation in Embedded, High-Performance and Cloud Computing*. River Publishers, 2016.

[13] R. Liu, X. Xie, K. Yu, and Q. Hu, "A survey on simulation optimization for the manufacturing system operation," *International Journal of Modelling and Simulation*, vol. 38, no. 2, pp. 116–127, 2018. [Online]. Available: https://doi.org/10.1080/02286203.2017.1401418

[14] I. Dmitry, *Operations and supply chain simulation with AnyLogic: Decision-oriented introductory notes for master students.*, 2nd ed. Berlin School of Economics and Law (preprint), 2017. [Online]. Available: https://www.anylogic.com/upload/pdf/Ivanov_AL_book_2017.pdf

[15] A. Klemmt, S. Horn, G. Weigert, and K.-J. Wolter, "Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 6, pp. 917 – 925, 2009, 18th International Conference on Flexible Automation and Intelligent Manufacturing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584509000362

[16] M. Zhou, Y. Pan, Z. Chen, and W. Yang, "Optimizing green production strategies: An integrated approach," *Computers & Industrial Engineering*, vol. 65, no. 3, pp. 517 – 528, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835213000715

[17] H. Zisgen, I. Meents, B. R. Wheeler, and T. Hanschke, "A queueing network based system to model capacity and cycle time for semiconductor fabrication," in *2008 Winter Simulation Conference*, Dec 2008, pp. 2067–2074.

[18] E. Valentin, A. Verbraeck, and H. Sol, "Advantages and disadvantages of building blocks in simulation studies," in *Simulation in Industry 15th European Simulation Symposium*. SCS-European Publishing House, 2003, pp. 141–148.

[19] R. Briesemeister and A. G. N. Novaes, "Comparing an approximate queuing approach with simulation for the solution of a cross-docking problem," *Journal of Applied Mathematics*, vol. 2017, p. 11, 2017.

[20] G. Shao, A. Brodsky, and R. Miller, "Modeling and optimization of manufacturing process performance using modelica graphical representation and process analytics formalism," *Journal of Intelligent Manufacturing*, vol. 29, no. 6, pp. 1287–1301, 2018.

[21] A. E. Kiasari, A. Jantsch, and Z. Lu, "Mathematical formalisms for performance evaluation of networks-on-chip," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 38:1–38:41, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2480741.2480755

[22] V. Nguyen and H. Bao, "An efficient solution to the mixed shop scheduling problem using a modified genetic algorithm," *Procedia Computer Science*, vol. 95, pp. 475 – 482, 2016, complex Adaptive Systems Los Angeles, CA November 2-4, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916324978

[23] R. L. Cruz, "A calculus for network delay. i. network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, Jan 1991.

[24] C. A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, and M. Fahl, "State-of-the-art review of optimization methods for short-term scheduling of batch processes," *Computers & Chemical Engineering*, vol. 30, no. 6-7, pp. 913–946, 2006.

[25] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983. [Online]. Available: http://doi.acm.org/10.1145/182.358434